

DOCKET NO.: 05-03-010 (UGSC01-05022)
Customer No. 45113

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of : Graham Hughes, et al.

Serial No. : 10/706,848

Filed : November 12, 2003

For : SYSTEM, METHOD, AND COMPUTER PROGRAM
PRODUCT FOR DISTRIBUTED TESTING OF PROGRAM
CODE

Group No. : 2192

Examiner : Eric B. Kiss

Conf. No. : 6082

MAIL STOP APPEAL BRIEF - PATENTS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLY BRIEF

Sir:

Applicants herewith respectfully submit that the Examiner's decision of March 21, 2008, finally rejecting Claims 1-21 in the present application, should be reversed, in view of the following arguments and authorities. This Reply Brief is submitted in response to the Examiner's Answer mailed December 22, 2008. No fee is believed due, but please charge any additional necessary fees to Deposit Account No. 50-0208.

TABLE OF CONTENTS

Real Party in Interest.....	3
Related Appeals or Interferences	4
Status of Claims	5
Summary of Claimed Subject Matter.....	7
Status of Amendments after Final.....	6
Grounds of Rejection to be Reviewed on Appeal.....	8
1. Are Claims 1-21 unpatentable under 35 U.S.C. § 103(a) over U.S. Patent Nos. 6,112,225 (Kraft <i>et al.</i>) and 6,360,268 (Silva <i>et al.</i>)?	8
ARGUMENT	9
Stated Grounds of Rejection	9
Legal Standards.....	10
First Ground of Rejection on Appeal.....	11
Grouping of Claims.....	22
REQUESTED RELIEF	23

APPENDIX A - Text of Claims on Appeal

APPENDIX B - Copy of Formal Drawings

APPENDIX C - Evidence Appendix

APPENDIX D - Related Proceedings Appendix

Real Party in Interest

The real party in interest, and assignee of this case, is Siemens Product Lifecycle Management Software Inc.

Related Appeals or Interferences

To the best knowledge and belief of the undersigned attorney, there are none.

Status of Claims

Claims 1-21 are under final rejection, and are each appealed.

Status of Amendments after Final

Claims 8 and 11 were amended after final rejection to correct informalities, and this amendment was entered. No other claims were amended after final rejection.

Summary of Claimed Subject Matter

The following summary refers to disclosed embodiments and their advantages, but does not delimit any of the claimed inventions.

A summary of claimed subject matter and support for independent claims was presented in the Appeal Brief, and is incorporated by reference.

Grounds of Rejection to be Reviewed on Appeal

1. Are Claims 1-21 unpatentable under 35 U.S.C. § 103(a) over U.S. Patent Nos. 6,112,225
(Kraft *et al.*) and 6,360,268 (Silva *et al.*)?

The Examiner has withdrawn the indefiniteness rejection under 35 U.S.C. § 112, second paragraph to claims 8-10. As Appellant specifically indicated in the Appeal Brief, the indefiniteness rejection under 35 U.S.C. § 112, second paragraph to claims 11-14 is not appealed, as claim 11 includes an obvious error overlooked by Appellant in the after-final amendment. As indicated in the Appeal Brief, Appellant will be happy to correct this error either by direct amendment or Examiner's amendment when proceedings on this appeal are concluded.

As no 35 U.S.C. § 112, second paragraph rejections remain appealed herein, this ground of rejection stated in the final Office Action and in the Appeal Brief is removed above.

ARGUMENT

Stated Grounds of Rejection

The rejections outstanding against the Claims are as follows:

1. In the March 21, 2008 Office Action, Claims 8-14 were rejected under 35 U.S.C. § 112, second paragraph, as indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as his invention. The indefiniteness rejection of claims 8-10 were withdrawn by the Examiner, and the indefiniteness rejection of claims 11-14 are not appealed. This ground of rejection is therefore not involved in the appeal.

2. In the March 21, 2008 Office Action, Claims 1-21 were rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent Nos. 6,112,225 (Kraft *et al.*) and 6,360,268 (Silva *et al.*).

Legal Standards

Relevant legal standards were addressed in the Appeal Brief, and are incorporated by reference.

First Ground of Rejection on Appeal

Claims 1-21 were rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent Nos. 6,112,225 (Kraft *et al.*, hereinafter “Kraft”) and 6,360,268 (Silva *et al.*, hereinafter “Silva”).

Claims 1-3, 8-10, and 15-17

These claims may be considered together *for this ground of rejection*.

Independent claim 1 describes

A method for testing code, comprising:

receiving a test request;

sending executable program code, corresponding to the test request, to a
client system;

receiving a response from the client system indicating that the client system
will perform a test, and indicating that the client system was not being
actively used when the executable program code was sent.

Independent claim 8 is drawn to a data processing system particularly configured to perform similar functions as the method of claim 1, and independent claims 15 is drawn to a computer program product with instructions for performing similar functions as the method of claim 1.

Claim 1 requires receiving a test request. The Examiner argues that this is taught by Kraft at col. 9, lines 1-27. This portion of Kraft was reproduced in the Appeal Brief. Claim 1 also requires sending executable program code, corresponding to the test request, to a client system. Kraft appears

to teach that a subtask is then received by peripheral computer 106, which may have been sent by the coordinating computer 102.

When step 608 finds that the host peripheral computer 106 is idle, the screen saver 204 initiates its screen saver in step 609 to prevent damage to the peripheral computer's monitor; this is only an option, however, and the screen saver function may be omitted entirely. More important, the screen saver 204 in step 609 also sends a message to the task manager 206 indicating that the peripheral computer 106 is idle. In response, the task manger 206 in step 610 determines whether the task execution engine 208 is already engaged in computation of an uncompleted subtask. If so, the task manager 206 retrieves the intermediate results from the buffer 210 (step 616), and directs the task execution engine 208 to continue executing the subtask (step 618). In another embodiment, steps 610 and 616 may be omitted; in this case, uncompleted subtasks interrupted by the user's applications or other activity are aborted and restarted when the next idle period begins.

Alternatively, if step 610 finds no subtask already in progress, the task manager 206 in step 612 requests a subtask. This involves submitting a subtask request to the coordinating computer 102. To benefit the coordinating computer 102, the subtask request may be accompanied by a machine-readable description of the peripheral computer's hardware components, operating system, and the like. Upon receipt of the subtask (step 614), the peripheral computer 106 has "subscribed" to the coordinating computer's aggregate task. At this point, the task execution engine 208 may start computing the subtask in step 618.

Kraft clearly does not teach or suggest receiving a test request at all. Kraft does teach that a subtask request is sent to coordinating computer 102 (which, of course, receives it).

The Examiner responds that Appellant "misconstrues" the rejection, and states that "Kraft is not relied upon as teaching receiving a *test* request, but rather receiving a subtask as part of a distributed processing system. ... Silva teaches that such distributed subtasks can be *test* requests as art of a distributed testing system." On the contrary, Appellant clearly understands that the Examiner is attempting to avoid showing the entire limitation in any reference, or showing any reference that teaches "receiving a test request", as claimed.

The claim requires "receiving a test request" and "sending executable program code, corresponding to the test request". This is not taught by any reference. No reference shows sending

Reply Brief – Serial No. 10/706,848 *Page 13*

executable code corresponding to a test request, and the Examiner attempts to avoid the plain language of the claims by simply assuming that the executable “subtask” can be some sort of testing. Kraft teaches sending a “subtask” in response to a request for a subtask, without regard to any testing. Far from misconstruing the rejection, Appellant notes that the Examiner fails to show the claim limitation in any reference or in the proposed combination of references.

Claim 1 also requires receiving a response from the client system *indicating that the client system will perform a test, and indicating that the client system was not being actively used when the executable program code was sent*. This is not taught or suggested by Kraft or by Silva; the Examiner does not allege any such teaching by Silva. Silva is only referenced for the concept of distributed testing.

For this limitation, the Examiner refers again to the passage reproduced in the Appeal Brief, but it is clear that Kraft’s coordinating computer 102 does not receive any sort of response indicating that the peripheral computer will perform a subtask, nor does coordinating computer 102 receive any sort of response indicating that the peripheral computer was not being actively used when the subtask was sent.

The Examiner is correct in that task requests and results are sent when the system is idle – but this is not the claim limitation. There is no response sent from the client to server indicating that the client will perform a task that it has been sent, as claimed – there appears to be no confirmation at all that a subtask has been received, or that the client will execute it. Nor is there any response sent from the client to server indicating that the client was not being actively used *when the subtask was*

sent – it may have been idle when the task was originally requested, and it may be idle again at some point when the task is completed, but there is no indication that the client is idle when the subtask is received.

This can be important, of course, when the server must determine how soon a task is to be completed – in Kraft’s system, there is no indication at all of when the task may be executed or completed. On the other hand, the claimed method provides that when the code *is received* by the client, it responds with an indication that it was idle on receipt and it will execute the task (perform the test). This is significantly different than the system of Kraft, and is not taught or suggested by Kraft.

The Examiner responds in the Reply Brief that there is no requirement of *when* various communications are sent, and that the claims only indicate an “eventual testing” that is confirmed when the test results are returned. This is a misreading of the plain language of the claims. The claims require a response that indicates that the client system *will perform* a test, clearly prospective language, and indicating that the client system *was not being actively used* when the executable program code was sent. The response clearly is sent after the program code is sent to the client, but before the test is performed.

The Examiner further argues that “the requesting of new tasks by the client of craft is another indication that the client will perform another task”. This is not a “response”, and is a request for a “task”, not an indication that a task will be performed. It is certainly not an indication that a *test* will be performed, which is the actual claim limitation.

As a whole, the Examiner appears to ignore the actual limitations of the claims, and instead attempts to “reinterpret” them to something that conforms to Kraft. The Examiner fails to show that the actual claim limitations are taught by any reference or combination of references at all.

Kraft does not teach or suggest this limitation of claim 1, or similar limitations of claims 8 and 15. Nor does Silva, nor does the Examiner allege any such teaching in Silva. As no art of record teach or suggest this limitation, no combination of the cited art can teach or suggest this feature. Nor is there any teaching or suggestion in the art to modify the references to meet this limitation, nor is there any teaching that would indicate that such a modification, if attempted would meet with predictable or successful results.

The obviousness rejections of these claims, and their dependents, should be reversed.

Claims 4, 6-7, 11, 13-14, and 18, 20-21

These claims may be considered together *for this ground of rejection*.

Independent claim 4 describes

A method for testing code in a client data processing system, comprising:
receiving executable code from a server system in a client data processing
system;
if the client data processing system is in an idle state when the executable
code is received, then
sending a response to the server system,
testing at least a portion of the executable code, and
sending test results to the server system.

Independent claim 11 is drawn to a data processing system particularly configured to perform similar functions as the method of claim 4, and independent claims 18 is drawn to a computer program product with instructions for performing similar functions as the method of claim 4.

The issues here are similar to those discussed above with regard to independent claims 1, 8, and 15. Claim 4 requires receiving executable code from a server system in a client data processing system. The Examiner again refers to Kraft's col. 9, lines 1-27, reproduced above. Kraft does teach that a "subtask" is received by peripheral computer 106. Though not specified by Kraft, it may be from coordinating computer 102.

Claim 4 also requires if the client data processing system is in an idle state *when the executable code is received*, then sending a response to the server system, testing at least a portion of the executable code, and sending test results to the server system. The Examiner again refers to Kraft's col. 9, but Kraft does not teach or suggest this limitation.

In Kraft's system, there is no response sent to the coordinating computer upon receipt of the subtask, whether or not the peripheral computer is idle when the subtask is received. Nor does Silva teach this limitation.

Contrast the limitation of Claim 5, discussed below, which requires that if the client data processing system is not in an idle state when the executable code is received, then no response is sent to the server and no testing is performed. This is also contrary to Kraft, which does not address the state of the client when the subtask is received, but will certainly execute the task and send a result whenever it does become idle.

The Examiner does not even address this clear distinction in his response in the Examiner's Answer, perhaps because neither Kraft nor Silva describe anything as contingent upon the client data processing system being in an idle state when the executable code is received, as claimed.

None of the cited references teach or suggest this limitation of claim 4, alone or in combination, or the similar limitations of claims 11 and 18. The rejections, therefore, of claims 4-7, 11-14, and 18-21 should be reversed.

Claims 5, 12, and 19

These claims may be considered together *for this ground of rejection*.

Claim 5 requires that if the client data processing system is not in an idle state when the executable code is received, then no response is sent to the server and no testing is performed.

Claim 12 is drawn to a data processing system particularly configured to perform similar functions as the method of claim 5, and claims 19 is drawn to a computer program product with instructions for performing similar functions as the method of claim 5.

These claims depend from claims 4, 11, and 18, respectively, and so the arguments above with respect to those claims apply here as well, and are hereby incorporated by reference.

These claims require that if the client data processing system is not in an idle state when the executable code is received, then no response is sent to the server and no testing is performed. Neither Kraft nor Silva discuss at all what happens if the client system is not in an idle state *when the executable code is received*, and they certainly don't teach this feature.

The Examiner refers to Kraft's col. 9, lines 36-55, which describe that subtask computation is "suppressed" when the processor is not idle. This passage does not address what happens *when the executable code is received*, as claimed.

On the contrary, Kraft specifically describes that the subtask is requested and received by the peripheral computer *when it is idle*, in the passage from col. 9 reproduced above with regard to claim

1. Kraft teaches that "When step 608 finds that the host peripheral computer 106 is idle, ... the

screen saver 204 in step 609 also sends a message to the task manager 206 indicating that the peripheral computer 106 is idle. In response, the task manger 206 in step 610 determines whether the task execution engine 208 is already engaged in computation of an uncompleted subtask. ... if step 610 finds no subtask already in progress, the task manager 206 in step 612 requests a subtask. This involves submitting a subtask request to the coordinating computer 102. ... Upon receipt of the subtask (step 614), ... the task execution engine 208 may start computing the subtask in step 618” (passage in full is above; this is edited to the relevant description).

It is clear that the subtask is requested and received by the peripheral computer *when it is idle*, and that the peripheral computer may start executing it immediately. The interrupt handling section reproduced above indicates that this processing is *paused* when the system is not idle, but Kraft clearly describes that after the pause, the subtask execution resumes at the next idle time, and the computation results are eventually sent to the coordinating computer.

Kraft does not contemplate, teach, or suggest, that if the client data processing system is not in an idle state when the executable code is received, then no response is sent to the server and no testing is performed. On the contrary, Kraft’s system only requests tasks when it is idle, and *all received tasks are eventually executed with results being sent to the server*. This is opposite that which is required by claims 5, 12, and 19.

Kraft’s system requests a subtask when idle, but may receive it while being actively used, process it at a later idle time, and will certainly then send the results to the server. This clearly does not meet the claim limitations.

The rejections of these claims should be reversed.

All obviousness rejections should be reversed.

Grouping of Claims

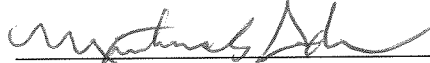
The claims on appeal do not stand or fall together, as may be seen from the arguments set forth above. Each claim or group of claims that has been argued separately under a separate subheading should be considered separately. While the appellant recognizes that a formal statement regarding the grouping of claims is no longer required, each claim should be considered separately; or at the very least each claim which is argued separately in the preceding sections of this brief should be considered separately.

REQUESTED RELIEF

The Board is respectfully requested to reverse the outstanding rejections and return this application to the Examiner for allowance.

Respectfully submitted,
MUNCK CARTER, LLP

Date: 02-04-09


Matthew S. Anderson
Registration No. 39,093

P.O. Drawer 800889
Dallas, Texas 75380
(972) 628-3600 (main number)
(972) 628-3616 (fax)
E-mail: manderson@munckcarter.com

ATTORNEY FOR APPELLANT

APPENDIX A -
Claims Appendix

1. (Original) A method for testing code, comprising:

receiving a test request;

sending executable program code, corresponding to the test request, to a client system;

receiving a response from the client system indicating that the client system will perform a test, and indicating that the client system was not being actively used when the executable program code was sent.

2. (Original) The method of claim 1, wherein executable program code, corresponding to the test request, is sent to multiple client systems.

3. (Original) The method of claim 1, further comprising retrieving a list of client system identifiers, the client system identifiers indicating client systems to which executable program code can be sent for testing.

4. (Original) A method for testing code in a client data processing system, comprising:

receiving executable code from a server system in a client data processing system;
if the client data processing system is in an idle state when the executable code is received,
then
sending a response to the server system,
testing at least a portion of the executable code, and
sending test results to the server system.

5. (Original) The method of claim 4, wherein if the client data processing system is not in an idle state when the executable code is received, then no response is sent to the server and no testing is performed.

6. (Original) The method of claim 4, wherein the testing is a coverage analysis test.

7. (Original) The method of claim 4, wherein the client data processing system is in an idle state when no user is actively operating the client data processing system.

8. (Previously Presented) A data processing system comprising a processor and accessible memory, the data processing system configured to:

receive a test request;
send executable program code, corresponding to the test request, to a client system; and
receive a response from the client system indicating that the client system will perform a test, and indicating that the client system was not being actively used when the executable program code was sent.

9. (Original) The data processing system of claim 8, wherein executable program code, corresponding to the test request, is sent to multiple client systems.

10. (Previously Presented) The data processing system of claim 8, wherein the data processing system is further configured to retrieve a list of client system identifiers, the client system identifiers indicating client systems to which executable program code can be sent for testing.

11. (Previously Presented) A data processing system comprising a processor and accessible memory, the data processing system configured to:

executable code from a server system in a client data processing system; and,

if the client data processing system is in an idle state when the executable code is received,

send a response to the server system,

test at least a portion of the executable code, and

send test results to the server system.

12. (Original) The data processing system of claim 11, wherein if the client data processing system is not in an idle state when the executable code is received, then no response is sent to the server and no testing is performed.

13. (Original) The data processing system of claim 11, wherein the testing is a coverage analysis test.

14. (Original) The data processing system of claim 11, wherein the client data processing system is in an idle state when no user is actively operating the client data processing system.

15. (Original) A computer program product tangibly embodied in a machine-readable medium, comprising:

instructions for receiving a test request;

instructions for sending executable program code, corresponding to the test request, to a client system;

instructions for receiving a response from the client system indicating that the client system will perform a test, and indicating that the client system was not being actively used when the executable program code was sent.

16. (Original) The computer program product of claim 15, wherein executable program code, corresponding to the test request, is sent to multiple client systems.

17. (Original) The computer program product of claim 15, further comprising instructions for retrieving a list of client system identifiers, the client system identifiers indicating client systems to which executable program code can be sent for testing.

18. (Original) A computer program product tangibly embodied in a machine-readable medium, comprising:

instructions for receiving executable code from a server system in a client data processing system;

instructions for, if the client data processing system is in an idle state when the executable code is received,

sending a response to the server system,

testing at least a portion of the executable code, and

sending test results to the server system.

19. (Original) The computer program product of claim 18, wherein if the client data processing system is not in an idle state when the executable code is received, then no response is sent to the server and no testing is performed.

20. (Original) The computer program product of claim 18, wherein the testing is a coverage analysis test.

21. (Original) The computer program product of claim 18, wherein the client data processing system is in an idle state when no user is actively operating the client data processing system.

DOCKET NO.: 05-03-010 (UGSC01-05022)

PATENT

APPENDIX B -
Copy of Formal Drawings

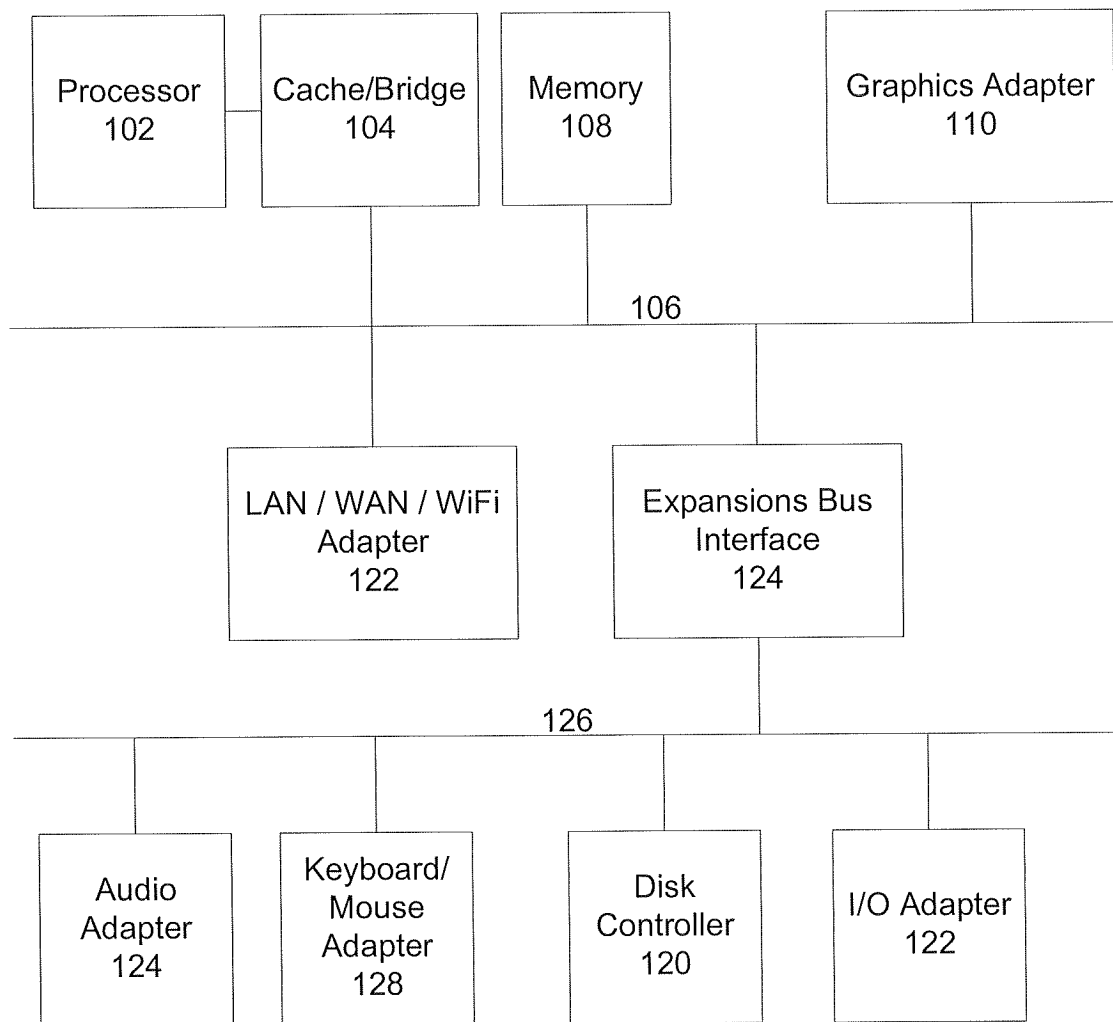


Figure 1

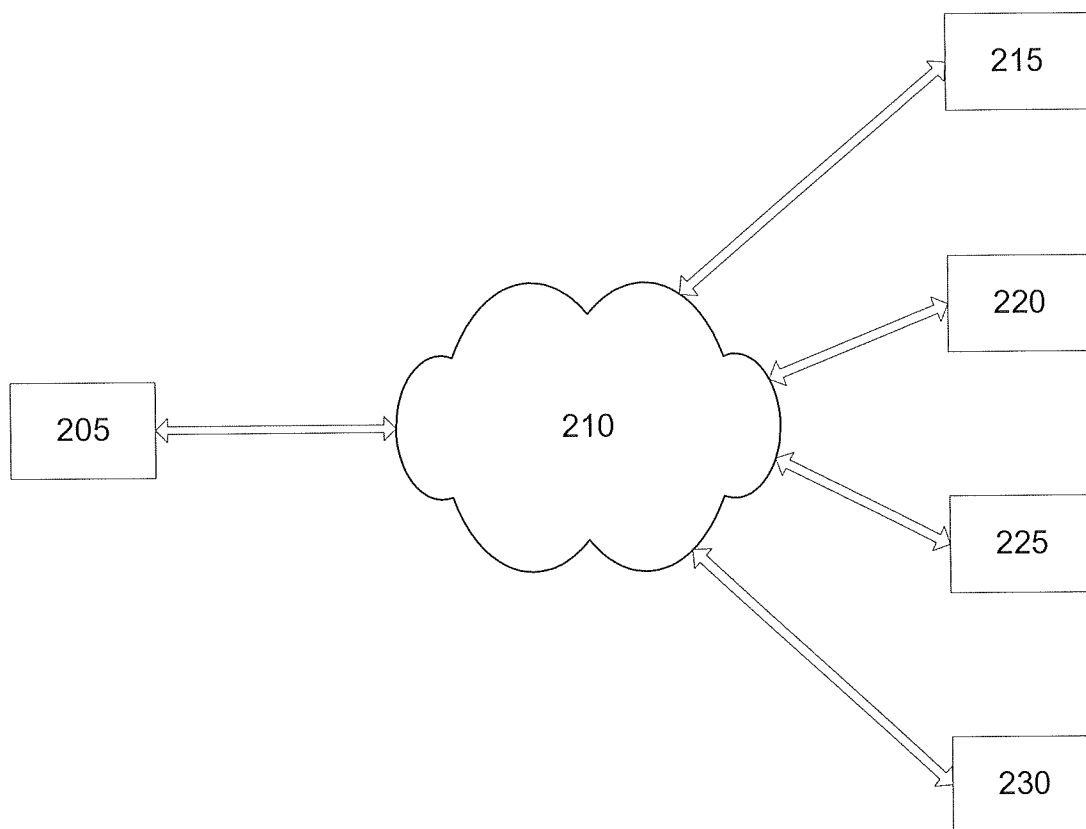


Figure 2

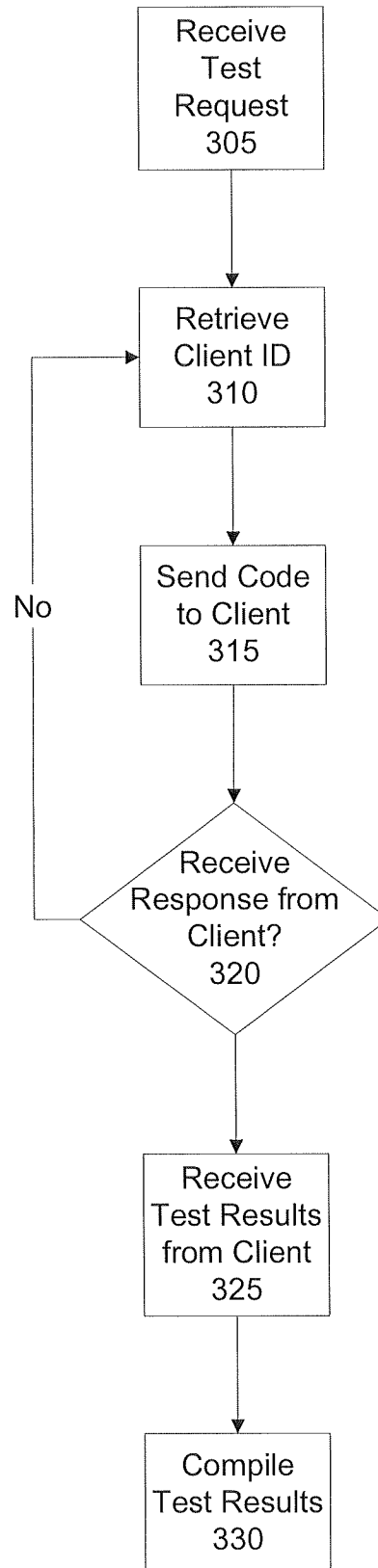


Figure 3

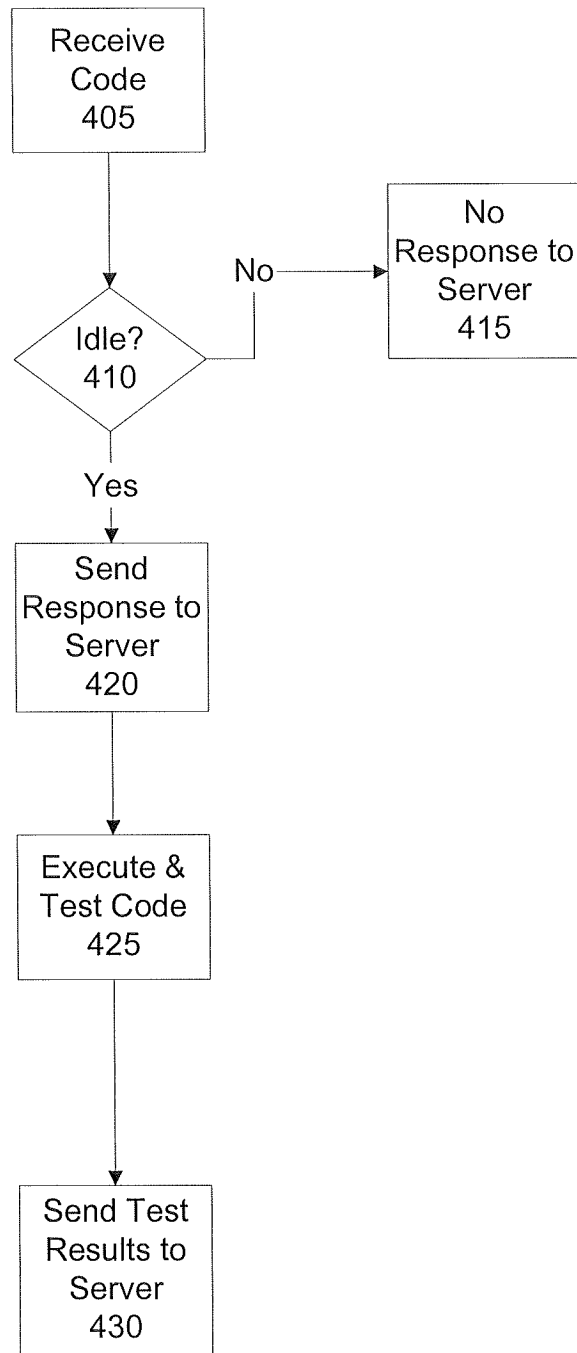


Figure 4

APPENDIX C -
Evidence Appendix

Not Applicable -- No other evidence was entered.

APPENDIX D -
Related Proceedings Appendix

Not Applicable -- To the best knowledge and belief of the undersigned attorney, there are none.